ZFW
AP
5

APPEAL BRIEF TRANSMITTAL LETTER

November 7, 2005

MAIL STOP APPEAL BRIEF - PATENTS
COMMISSIONER FOR PATENTS
P.O. Box 1450
ALEXANDRIA, VA 22313-1450

Re: Appellants: Blightman, et al.
Assignee: Alacritech, Inc.
Title: "Network Interface Device Employing A DMA Command Queue"
Serial No.: 09/855,979 Filing Date: May 14, 2001
Examiner: Pierre-Michel Bataille
Atty. Docket No.: ALA-016 Art Unit: 2186

Dear Sir:

Transmitted herewith are the following documents:
(1) Appeal Brief (25 pages);
(2) Return Postcard;
(3) A check for the Appeal Brief Filing Fee ($500); and
(4) This transmittal sheet.

☐ No additional Fee is required.
☒ The fee has been calculated as shown below:

Fees						
	REMAINING AFTER AMENDMENT		HIGHEST NO. PREVIOUSLY PAID FOR	EXTRA CLAIMS PRESENT	RATE	ADDITIONAL FEE
TOTAL CLAIMS	34	minus	34	0	$50	$0.00
INDEP. CLAIMS	8	minus	8	0	$200	$0.00
Total Additional Claim Fee						$0.00
Fee for Extension of Time (__ month)						$0.00
Fee for Appeal Brief (check enclosed).						$500.00
					TOTAL	$500.00

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

By LesterWallace
T. Lester Wallace

Date of Deposit: November 7, 2005

Respectfully submitted,

LesterWallace

T. Lester Wallace
Attorney for Appellants
Reg. No. 34,748

Applicants: Blightman et al.

Assignee: Alacritech, Inc.

Title: "Network Interface Device Employing A DMA Command Queue"

Appl. No.: 09/855,979 Filing Date: May 14, 2001

Examiner: Pierre-Michel Bataille Art Unit: 2186

Docket No.: ALA-016

November 7, 2005

Mail Stop Appeal Brief - Patents
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

This Appeal Brief is filed under 37 CFR §41.37 in support of the appeal noticed September 6, 2005.

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Alacritech, Inc., as named in the caption above.

II. RELATED APPEALS AND INTERFERENCES

Based on information and belief, there are no appeals or interferences that could directly affect or be directly affected by or have a bearing on the decision by the Board of Patent Appeals and Interferences (the "Board") in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-34 are pending. Claims 5 and 25-27 stand allowed. Claims 1-4, 6-24 and 28-34 are rejected under §102 over the AAPA (Applicants' Admitted Prior Art) of Figure 1. The §102 rejection of Claims 1-4, 6-24 and 28-34 is being appealed. The "Claims Appendix" below lists the claims that are the subject of this appeal.

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER[1]

A summary explanation of the claimed subject matter is provided by: 1) first discussing the so-called "AAPA" (Applicants' Admitted Prior Art) illustrated in Figure 1 of Applicants' specification, and 2) then by discussing the specific embodiment illustrated in Figures 2 and 3 of Applicants' specification.

SUMMARY OF THE AAPA: Figure 1 (replicated below) illustrates the "network interface device" (NID) 100 of the AAPA.

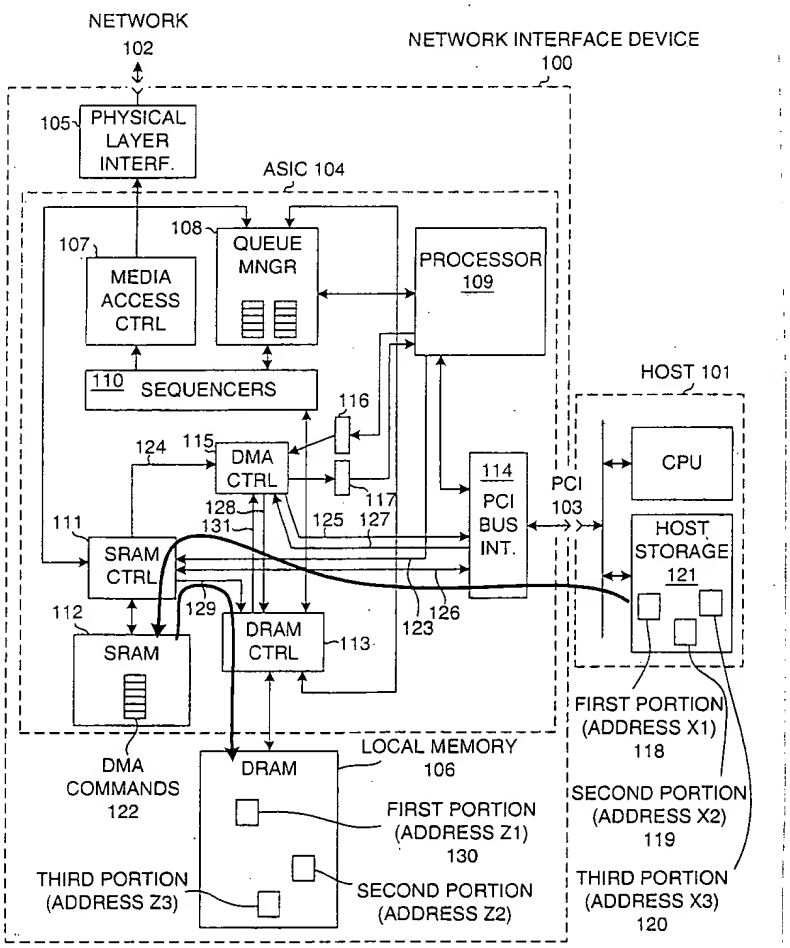

FIG. 1
(PRIOR ART)

[1] The following summary pursuant to 37 CFR §41.37(c)(1)(v) is a concise explanation of the claimed subject matter. Aspects of specific embodiments that are mentioned in the explanation are not to be imported into the claims. This summary does not limit the claims.

Prior Art NID 100 is an expansion card that is coupled to a host computer's PCI bus 103. NID 100 couples the host computer 101 to a network 102. (2: 3-7). In one example, NID 100 is usable to retrieve three portions of data 118-120 from host storage 121 and to output them to network 102 in the form of a data payload of a packet. (2:19-22). The background section explains that all three portions of data 118-120 are to be present in local memory 106 before packet transmission begins. (2:25-26). To move the three portions 118-120 into local memory 106, the processor 109 on NID 100 places DMA commands 122 into SRAM 112. (2:27-3:8). DMA controller 115 then executes the DMA commands. When DMA controller 115 completes execution of a DMA command, it sets a corresponding bit in a 32-bit DMA command complete register 117. Processor 109 monitors the bits in the DMA command complete register 117 to determine when DMA commands have been completed. (3:8-12).

The background section explains that "***DMA controller 115 may execute DMA commands in an order different from the order in which the DMA commands were placed in SRAM 112 by processor 109***." (emphasis added) (4:12-15). The background section explains that while DMA controller 115 is moving the first portion 118 to local memory 106, the processor 109 may place additional DMA commands into SRAM 112. When DMA controller 115 finishes executes the DMA command for moving first portion 118, the DMA controller 115 may next execute the DMA command for moving the third portion 120. (4:15-20).

The background section explains that because all three data portions 118-120 are to be in local memory 106 before transmission of the packet begins, processor 109 "***cannot only check that the last move in the sequence is completed***." (emphasis added)(4:22-23). Rather, processor 109 must check to make sure that all the moves are completed before processor 109 can go on in its software to execute the instructions that cause the ultimate packet to be formed and output from NID 100. (4:23-27).

SUMMARY OF THE CLAIMED SUBJECT MATTER: Figure 2 (replicated below) illustrates a novel "network interface device" (NID) 200.

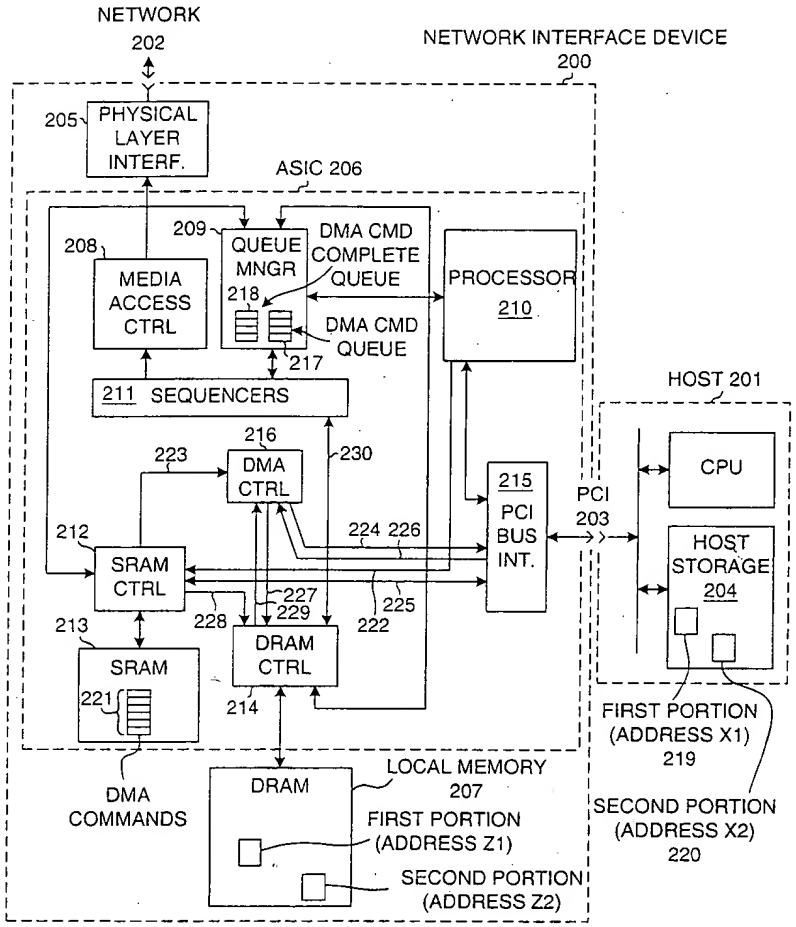

FIG. 2

Network interface device (NID) 200 includes "queue manager hardware" 209 that maintains a "DMA command queue" 217 as well as a "DMA command complete queue" 218. (9:4-5). Values can be pushed onto these queues, and values can be popped off of these queues. (9:19-24). There are DMA commands 221 in SRAM 213. In one example, a value on the "DMA command queue" 217 is a pointer to a corresponding one of the DMA commands 221 in SRAM 213. (10:27; 11:28; claims 7 and 8). In one example, the values on "DMA command complete queue" are "termination values". Each termination value is indicative of a corresponding one of the DMA commands. (12:14-18; 11-13).

Operation of NID 200 is explained in connection with the method set forth in the flowchart of Figure 3 (Fig. 3). There are two portions of data 219 and 220

in host storage 204 (see Fig. 2). Both of these two portions of data are to be present in local memory 207 on NID 200 before processor 210 causes the two portions of data to be output from the NID in the form of a data payload of a TCP/IP packet (9:30-10:8).

To move the two portions of data, processor 210 puts a DMA command to move a portion of data from host storage 204 to local memory 207 in SRAM 213, and causes a corresponding value to be pushed onto DMA command queue 217. The value is popped off the DMA command queue 217 and is executed by the DMA controller 216 such that the portion of data is moved from host storage 204 to local memory 207. When the DMA command has been completed, the DMA controller 216 causes a termination value to be pushed onto the DMA command complete queue 218. Unlike the prior art AAPA of Figure 1, the "***DMA controller on the network interface device executes DMA commands in the order in which the processor pushed the associated values onto the DMA command queue***." (5:13-16). These steps of pushing values onto the DMA command queue and popping the values off and executing corresponding DMA commands is repeated (see Fig. 3: step 302) for each of the portions of data that need to be moved.

As explained on page 13 of the specification, "processor 210 need not monitor the complete status of all the DMA commands involved as in the prior art example of Figure 1." (13:24-26). Rather, processor 210 pops the "DMA command complete queue" 218. When the value popped off the DMA command complete queue is the termination value for the last DMA command to be executed to move the necessary portions of data (in this case, the DMA command for second portion 220), then it is known that all the preceding DMA commands have been executed. (14:3-7). Processor 210 then executes instructions that cause the first and second data portions 219 and 220 to be output from NID 200 in the form of a data payload of a network communication (14:11-18; Fig. 3, step 303). The first and second data portions 219 and 220 are output by media access control circuitry 208 and a physical layer interface 205. (Fig. 2, 208 and 205; Claim 11).

Independent Claim 28 contains a "means plus function" element and therefore is specifically addressed pursuant to §41.37. Independent Claim 28 recites a "means for maintaining a DMA command queue..." As is evident from dependent Claim 29, the scope of the means element in Claim 28 comprises a processor such as processor 210. The instructions executed by the processor 210 to perform the recited function may or may not be part of the recited means. The scope of the means element of independent Claim 28 also comprises a software queue mechanism for maintaining the recited DMA command queue (6:27-31). As is evident from dependent Claim 30, the scope of the means element in Claim 30 further comprises a "hardware queue manager" such as hardware queue manager 209. For additional details on the structure and operation of one embodiment of a hardware queue manager, see U.S. Patent Application Serial No. 09/416,925, now U.S. Patent No. 6,470,415, the subject matter of which is incorporated by reference. As set forth in the present application, the NID functionality can be carried out on an expansion card, or can be embodied an integrated circuit that is disposed on the motherboard of the host computer. (15:4-30).

Dependent Claim 34 recites that the NID of independent Claim 31 is a "means for performing fast-path transport and network layer protocol processing". Disclosures of particular structures and methods for performing fast-path transport and network layer protocol processing are found in U.S. Patent Application 09/464,283, now U.S. Patent No. 6,427,173, the subject matter of which is incorporated into the present application. (8:20-31).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-4, 6-24 and 28-34 are rejected under 35 U.S.C. §102 as being anticipated by AAPA (Applicants' Admitted Prior Art).

VII. ARGUMENT

A. The §102 Rejection

Claims 1-4, 6-24, and 28-34 are rejected under 35 U.S.C. §102 for being anticipated by "Applicant Admitted Prior Art (AAPA)". This is the only §102 rejection, and there is no §103 rejection. The AAPA is the only prior art used to reject any claim in the instant application.

With respect to independent Claim 1, the Examiner maintains that "command queue 122 in SRAM 112" of the AAPA is the recited "DMA command queue" of Claim 1 (Final Office Action, page 6, lines 11-12). The Examiner states:

> With respect to claim 1, Applicant Background Prior Art teaches the invention as claimed, a method, comprising: maintaining on a network interface device a DMA command queue [NID 100 with ***command queue 122 in SRAM 112***]; processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command [processor 109 placing DMA commands Par. 0008]; popping a value off the DMA command queue [Par. 0005], a DMA controller on the network interface device then executing a DMA command indicated by the popped value [DMA CTRL 111 executing DMA Commands, Par. 0005]; repeating and such that a first portion of data is transferred from host storage to a local memory on the network interface device and such that a second portion of data is transferred from the host storage to the local memory on the network interface device [Par. 0007]; and after both the first portion and the second portion are present the local memoyr, outputting the first and second portions of data from the network interface device to a network, the first and second portions making up at least a part of a data payload of a network communication [Par. 000008]. (emphasis added).

Similarly, with respect to independent Claim 11, the Examiner maintains that "DMA command queue 122 in SRAM 112" of the AAPA is the recited "DMA command queue" of Claim 11 (Final Office Action, page 8, lines 12-13). The Examiner states:

> With respect to claims 11 and 34, Applicant's Background Art in Fig. 1 discloses the invention as claimed, a network interface device (100) comprising: queue manager hardware [QUEUE MNGR 108] that maintains a DMA command queue [***DMA command queue 122 in SRAM 122***]; a processor (109) coupled to the queue manager hardware (108), the processor causing values to be pushed onto the DMA command

> queue [Par. 0005]; DMA controller (115) coupled to the queue manager hardware (108), the DMA controller executing DMA commands, the DMA commands executed being indicated by values popped off the DMA command queue [Par. 0005]; local memory DRAM local memory 106] that temporarily stores a first portion data transferred by execution of one or more DMA commands from data storage on a host coupled to the network interface device into the local memory, the local memory (106) also temporarily storing a second portion of data transferred by execution of one or more DMA commands (DMA command 122) from the data storage (121) on the host to the local memory (106)[Par. 0006 & 0007]; and physical layer interface (105) and media access control circuitry (107) outputting the first and second portions data from the network interface device network, the first and second portions of data being output in the form of a data payload of a network communication [Par. 0004]. (emphasis added).

Similarly, with respect to independent Claims 21[2] and 28 and 31, the Examiner maintains that "DMA command queue 122" of the AAPA is the recited "DMA command queue" (Final Office Action, page 5, lines 7-8). The Examiner treats the three independent claims together, and states:

> With respect to claims 21 and 28-31, Applicant Background Art teaches the invention as claimed, an apparatus as described in Prior Art Fig. #1 and paragraphs 0003 to 0008, implementing the method comprising: a DMA command queue ***(DMA Commands 122)*** to ensure that a plurality of DMA moves are completed in a particular sequence (DMA controller may execute DMA commands in an order different from the order the DMA commands were placed), each of the DMA moves being a move of information from one location on a network interface device to another location on the network interface device (moves the first, second and third portions from host storage 121 to DRAM 106), the DMA command queue being maintained by queue manager hardware on the network interface device (Queue manager hardware 108 on the network interface device (NID) 100), the DMA moves being carried out by a DMA controller, the DMA controller being a part of the network interface device (DMA controller 115 being part of the network interface device (NID) 100); and outputting at least part of the information from the network interface device (transmission of data packet). (emphasis added).

Accordingly, the one and only §102 rejection that is the subject of this appeal depends on the "DMA COMMANDS 122" in the AAPA of Figure 1 being the recited "DMA command queue."

Applicants were somewhat bewildered with how the Examiner could consider the "DMA COMMANDS 122" in the AAPA of Figure 1 to be a "queue". Applicants' Amendment of March 2, 2005 therefore requested that the Examiner explain in a little more detail how it is that the Examiner could consider the DMA commands 122 to be a "queue".

The Examiner responded in the Final Office Action of June 3, 2005 by making two points of note. The two points are repeated here because they help under the §102 rejection.

First, the Examiner stated that the statement in the AAPA (Pa. 0008) that the DMA controller may execute DMA commands in an order different from the order in which the DMA commands were placed in SRAM 112 "implies a list of commands" where placed in a queue for fetching and execute in a particular order..." (Final Office Action, page 4, lines 3-8). Evidently, the Examiner concedes that a queue is not actually disclosed, but rather that the wording of the AAPA "implies" that a queue.

Second, the Examiner provided the following definition of the term "queue" (Final Office Action, page 3, lines 9-12) that the Examiner had evidently been using.

> Computer Science Dictionary defines queue as:
> a. A sequence of stored data or programs awaiting processing.
> b. A data structure from which the first item that can be retrieved is the one stored earliest.

After considering the definition used by the Examiner, Applicants were still bewildered with how the Examiner could consider the "DMA COMMANDS 122" in the AAPA of Figure 1 to meet the definition (either part a or part b). A telephonic interview was therefore conducted on August 4, 2005. Some of the conversation that transpired in the interview is repeated here because the Examiner's rationale is not recorded anywhere in the formal Final Office Action.

---

[2] The Office Action does not include a parallel description of independent Claim 14.

In the interview of August 4, 2005, the undersigned submitted that a "queue", as the term is used by Applicants in Applicants' specification and claims, involves an ordered list of items. The undersigned explained that the DMA commands 122 in the AAPA, in contrast, are just a bunch of items stored in a memory. They are not stored in an order. They can be pulled out and executed by DMA controller 15 in any order - a specific order is not enforced. The undersigned explained that just because items can be pulled out willy-nilly and happen to be pulled out in some order, does not mean that the items as stored are in a list, or an ordered list.

The Examiner disagreed. The Examiner stated that he could consider a group of DMA commands stored in a memory (such as SRAM 112) to be in a "queue," if those items could be pulled out and executed in some order, or in any order. The Examiner stated that as long as there is some order in which the DMA commands are pulled out in a particular instance, then in the Examiner's view, the definition of "queue" is met.

Applicant thereafter proposed adding words to the rejected claims that would make it explicit that the recited "DMA command queue" involves an ordered list of DMA commands. The Examiner considered the proposed amendments, but then stated that if the claims were so amended that they would still be anticipated by Figure 1 of the AAPA. The Examiner stated that he would not enter the proposed amendments, and that making the proposed amendments would have no point. The Examiner's rationale for his conclusion is unknown.

Applicants appeal the §102 rejection.

B. <u>There Is No "DMA Command Queue" in the AAPA</u>

There is no "DMA command queue" in the AAPA of Figure 1. The DMA commands 122 can be put into SRAM 112 in one order, and can be taken out in another order. The AAPA explains that

> "DMA controller 115 may execute DMA commands in an order different from the order in which the DMA commands were placed in SRAM 112 by processor 109. For example, while DMA controller 115 is moving the first portion 118 to DRAM 106, the processor 109 may place additional DMA commands into SRAM 112. When DMA controller 115 finishes moving

first portion 118, the DMA controller 115 may fetch a DMA command to move the third portion 120 next." (4:12-20).

Alternatively, the DMA commands 122 can be put into SRAM 112 in one order, and can be taken out in the same order. There is no ordered list of DMA commands stored in SRAM 112. The Examiner's assertion that the AAPA statement that DMA commands can be executed in an order different from the order in which the DMA commands were placed in SRAM 112 "implies a list of commands" is just an assertion by this Examiner. There is no such implication in paragraph 0008. The undersigned in fact wrote the statement in paragraph 0008, and the undersigned was not implying or trying to imply that there is any ordered list of DMA commands in SRAM 112. DMA commands 122 are just a bunch of DMA commands stored in SRAM 112.

In view of this, if the Examiner's own definition for "queue" is used, then it is clear that the DMA commands 122 are not a "queue" The Examiner's definition actually is two definitions. Each is discussed separately below.

The first Computer Science Dictionary definition of the term "queue" is: "a. a ***sequence*** of stored data or programs awaiting processing" (emphasis added). If the DMA commands 122 can be taken out of SRAM 112 and executed in any order, then there is no "sequence" of DMA commands that are stored. Because the DMA commands 122 are not in any sequence, the DMA commands 122 in the AAPA do not meet the Examiner's own definition a. for a queue.

The second Computer Science Dictionary definition of the term "queue" is similar to the definition of a FIFO (first-in-first-out). The definition is: "b. A data structure from which ***the first item*** that can be ***retrieved is the one stored earliest***." If the DMA commands 122 can be taken out and executed in any order, then it is not necessarily so that the first DMA command retrieved (and executed by DMA controller 115) will be the first DMA command that was stored. As set forth above, page 4, lines 12-14 of the AAPA specifically say that the DMA controller may execute DMA commands in an order different from the order in which the DMA commands were placed in SRAM 112. Because the DMA commands 122 in the AAPA can be taken out and executed in any order, the

DMA commands 122 in the AAPA do not meet the Examiner's own definition b. for a queue.

Because each of the appealed independent claims recites a "DMA command queue", and because there is no DMA command queue in the AAPA of Figure 1, the §102 rejection of Claims 1-4, 6-24, and 28-34 is improper. The Board is requested to reverse the improper §102 rejection.

C. The Examiner Does Not Even Use His Own Definition

As set forth above, the Examiner maintains that he can consider a group of DMA commands stored in a memory (such as SRAM 112) to be in a "queue," if those items can be pulled out and executed in some order, or in any order. The Examiner states that as long as there is some order in which the DMA commands are pulled out in a particular instance, then the definition of "queue" is met. (Telephonic Interview of August 4, 2005)

When the Examiner says this, the Examiner is not even using his own definition[3] of "queue". Definition a. of the term "queue" advanced by the Examiner is "a ***sequence*** of stored data or programs awaiting processing" (emphasis added). If the DMA commands can be pulled out in any order, then they are not being pulled out in "sequence" as required by the Examiner's own definition.

Definition b. of "queue" advanced by the Examiner is "a data structure from which the ***first item*** that ***can be retrieved*** is the one ***stored earliest***." If the definition b. would describe a data structure where any item can be pulled out at any time, then the identification of "earliest stored" item being the one that "can be retrieved" would be meaningless. Quite to the opposite of what the Examiner says, definition b. implies an ordered list, and more particularly a first-in-first-out (FIFO). Accordingly, when Examiner states that as long as there is some order in which the DMA commands are pulled out of SRAM 112 in a particular instance, then the definition of "queue" is met, the Examiner is not applying his own definition of "queue".

D. The Term "DMA Command Queue" Should Be Interpreted To Be Consistent With The Applicants' Use Of The Term

It would be improper and nonsensical to use a definition of the term "queue" that is inconsistent with the usage given to the term "queue" by Applicants. The term "DMA command queue" should be interpreted to be consistent with the way Applicants have used the term in their specification and in their claims. The way Applicants have used the term "queue" is therefore discussed.

The original claims were written with the AAPA clearly in mind. The AAPA is part of the application itself. Nowhere does the AAPA use the term "queue" to describe the DMA commands 122 in SRAM 112. Nowhere does the AAPA use the queue-related terms "push" and "pop" in connection with the placing of DMA commands into SRAM 112, or in connection with the fetching of DMA commands out of SRAM 112. In stark contrast, Applicants have used the terms "queue", "push" and "pop" to describe the "DMA command queue" in the specific embodiment of Figure 2.

If the specification and claims are interpreted properly, then the term "queue" should be given a meaning that: 1) describes DMA command queue 217 in Figure 2, and 2) excludes the DMA commands 122 in the AAPA. The Examiner's definition of the term "queue", to the extent it is inconsistent with Applicants' usage of the term "queue" and leads to an illogical conclusion that the original claims were drafted to cover the prior art, should not be used.

Applicants submit that the more logical interpretation of the term "queue" implies an ordered list. The "queue" of Claim 1 can, for example, be pushed and popped as recited by the claim language itself. In contrast, the DMA commands 122 in SRAM 112 of the AAPA are not in an ordered list. The DMA commands 122 can be removed from SRAM 112 in any order. The group of DMA commands 122 cannot be "pushed". The group of DMA commands 122 cannot be "popped". As such, Applicants do not consider the group of DMA commands

[3] The Computer Science Dictionary definition advanced by the Examiner is very short and

122 in SRAM 112 to be a "queue". This is evident from the specification and original claims.

E. The AAPA Does Not Disclose The Recited "Pushing" and "Popping"

One usage of the term "queue" implies an ordered list of items, where the list has a "head" and a "tail". Although it depends on the particular queue, an item can generally be "pushed" onto the "tail". If the queue is a last-in-first-out (LIFO) type, then the last item pushed onto the tail can be "popped" back off the "tail". If the queue is a first-in-first-out (FIFO) type, then the oldest item to be pushed onto the queue can be "popped" off the "head". In some queues, both the head and tail can be pushed and popped. Regardless of the particulars, however, the terms "push" and "pop" are used in connection with ordered lists (i.e., queues). As Applicants have used the terms in the specification, only the ends of the queue are "pushed" or "popped."

Consistent with this, independent Claims 1, 11, 14, 28 and 31 recite values either "pushed" or "to be pushed" or "pushing" values onto a "DMA command queue". There is no disclosure in the AAPA of DMA commands being "pushed". Claims 1, 11, 14, 28 and 31 are distinguished from the AAPA of Figure 1 for this additional reason.

The §102 rejection indicates that the statement "placing the DMA commands into SRAM 112" (AAPA, par. 0008) is a disclosure of the recited "pushed" or "pushing". It is, however, not true that the statement in paragraph 0008 of the AAPA is a disclosure of the recited "pushed" or "pushing". Applicants submit that DMA commands can be placed into a memory without "pushing" them onto any queue. That is in fact what is occurring in the AAPA of Figure 1. Accordingly, despite what the Examiner says, the "placing the DMA commands into SRAM 112" citation from the AAPA is simply not a disclosure of "pushing" anything onto a queue.

Independent Claims 1, 11, 14 and 31 also recite "popping" values off the DMA command queue or values "popped" of the DMA command queue. As in

---

Applicants submit, incomplete. Due to its being incomplete, it is to some extent inaccurate.

the case of "pushing", there is no disclosure in the AAPA of DMA commands 122 being "popped". Again, this is because there is no ordered list of DMA commands.

The §102 rejection asserts that there is a disclosure in paragraph 0005 of the AAPA of "popping". This is, however, not true. Paragraph 0005 says that DMA controller 115 can fetch and execute the DMA command, but nowhere is the act of popping disclosed. Nowhere is the term "pop" used. Applicants point out that a DMA command can be fetched from memory without "popping" it off any queue. That is in fact what is occurring in the AAPA of Figure 1.

Accordingly, despite what the Examiner says, the sections of the AAPA pointed to by the Examiner do not disclose "pushing" or "popping" DMA commands onto or off of any queue. The AAPA of Figure 1 therefore cannot anticipate Claims 1, 11, 14, 28 and 31 for this additional reason. The Board is requested to reverse the improper §102 rejection.

F. The AAPA Does Not Disclose Ensuring That DMA Moves Are "Completed In A Particular Sequence"

Independent Claim 21 recites a step of using a DMA command queue "to ensure that a plurality of DMA moves are completed in a particular sequence". (emphasis added). Independent Claim 28 recites a DMA controller that executes a plurality of DMA commands "such that the DMA commands are completed in a particular order". Independent Claim 31 recites popping the DMA command queue such that a DMA controller on the NID "executes the plurality of DMA commands in the order in which the associated values were pushed onto the DMA command queue".

In sharp contrast to this, the AAPA says that "DMA controller 115 may execute DMA commands in an order different from the order in which the DMA commands were placed in SRAM 112 by processor 109." (4:12-14). The Examiner has not identified anything in the AAPA that says that it is ensured that the DMA moves are completed in a particular sequence. The AAPA, therefore, does not disclose the specific recitations set forth above in Claims 21, 28 and 31.

The Board's reversal of the improper §102 rejection is requested.

G. The AAPA Does Not Disclose "Queue Manager Hardware" That "Maintains" A "DMA Command Queue"

Each of independent Claims 11, 14 and 21 recites that "queue manager hardware" maintains a DMA command queue.

The §102 rejection of independent Claim 11 states "queue manager hardware [QUEUE MNGR 108] that maintains a DMA command queue [DMA command queue 122 in SRAM 122][4]. The bracketed reference to "QUEUE MNGR 108" in the statement of the rejection is therefore a reference to the part of the AAPA that the Examiner maintains is the "queue manager hardware" that maintains the DMA command queue as recited in Claim 11.

The Final Office Action contains no statement of where the Examiner maintains the elements of Claim 14 are found in the AAPA of Figure 1.

The §102 rejection of independent Claim 21 states "the DMA command queue being maintained by queue manager hardware on the network interface device (Queue manager hardware 108 on the network interface device (NID) 100)".[5] The reference in the parentheses to "Queue manager hardware 108" is therefore a reference to the part of the AAPA that the Examiner maintains is the "queue manager hardware" that maintains the DMA command queue as recited in Claim 21.

Applicants' respectfully submit that the Examiner has no basis for his assertions that queue manager hardware 108 in the AAPA of Figure 1 maintains any DMA command queue. There is no such statement or disclosure in the AAPA. The queue manager hardware 108 of the AAPA of Figure 1 was used to maintain other queues. Nothing in the AAPA says that queue manager 108 has anything to do with the DMA commands 122 in SRAM 112.

For this additional reason, the §102 rejections of Claims 11, 14 and 21 is improper. The Board is requested to reverse the §102 rejection.

---

[4] Final Office Action, page 8, lines 12-13.

H. The AAPA Does Not Disclose A "DMA Command Complete Queue" As Recited In Claims 2 and 3

Dependent Claims 2 and 3 recite a "DMA command complete queue". The statement of the §102 rejection relating to Claim 2 (Final Office Action, page 7, lines 5-9) is as follows:

> "With respect to claim 2, Applicant Background Prior Art teaches maintaining a DMA command complete queue on the network interface device [DMA command complete queue 17], the DMA controller pushing values onto the DMA command complete queue, the processor popping values off the DMA command complete queue [Par. 0007]."

Similarly, the statement of the §102 rejection relating to Claim 3 (Final Office Action, page 7, lines 10-13) is as follows:

"With respect to claim 3, Applicant Background Art teaches the system and method wherein the processor uses the DMA command complete queue to determine that the first and the second portions of data are both present in local memory on the network interface device [DRAM local memory 106; Par. 0007]."

The §102 rejection therefore depends upon the "DMA command complete register 117" of the AAPA being the recited "DMA command complete queue" of Claims 2 and 3.

Despite the implication by the Examiner, reference numeral 117 in Figure 1 identifies a "DMA command complete ***register***", not a "DMA command complete ***queue***" as the Examiner says (emphasis added). The description of what the register 117 does is consistent with it being a register and not a queue. The AAPA explains that when DMA controller 115 executes a DMA command, the DMA controller 115 sets a corresponding bit in the DMA command complete register 117. The AAPA explains that DMA commands may be executed in an order different from the order in which then were placed into SRAM 112 by processor 109. The AAPA says that processor 109 monitors the DMA command complete register 117 to determine when the DMA command has been completed. The description of register 117 is not a description of a "queue". There is no ordered list in register 117. Register 117 is just a register, like the

[5] Final Office Action, page 5, lines 13-15.

AAPA says. There is no "DMA command complete queue" in the AAPA. Claims 2 and 3 that recite a "DMA command complete queue" therefore are not anticipated by the AAPA.

Because there is no "DMA command complete queue" in the AAPA, the Board is requested to reverse the improper §102 rejection of Claims 2 and 3.

VIII. CONCLUSION

For the numerous reasons set forth above, the AAPA of Figure 1 does not and cannot anticipate Claims 1-4, 6-24 and 28-34. There is simply no "DMA command queue" in the AAPA of Figure 1. There is no "pushing" and "popping" of DMA commands onto or off of any queue in the AAPA of Figure 1. There is no "queue manager hardware" that maintains any DMA command queue in the AAPA of Figure 1. There is no "DMA command complete queue" in the AAPA of Figure 1.

The Board is requested to reverse the improper §102 rejection of Claims 1-4, 6-24 and 28-34.

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

By 
T.Lester Wallace

Date of Deposit: November 7, 2005

Respectfully submitted,

T. Lester Wallace
Attorney for Appellants
Reg. No. 34,748

6601 Koll Center Pkwy, Ste 245
Pleasanton, CA 94566
Tel: (925) 485-9923
Fax: (925) 621-2119

IX. CLAIMS APPENDIX

1. A method, comprising:

(a) maintaining on a network interface device a DMA command queue;

(b) a processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command;

(c) popping a value off the DMA command queue, a DMA controller on the network interface device then executing a DMA command indicated by the popped value;

(d) repeating (b) and (c) such that a first portion of data is transferred from host storage to a local memory on the network interface device and such that a second portion of data is transferred from the host storage to the local memory on the network interface device; and

(e) after both the first portion and the second portion are present in the local memory, outputting the first and second portions of data from the network interface device to a network, the first and second portions making up at least a part of a data payload of a network communication.

2. The method of Claim 1, further comprising:

(f) maintaining a DMA command complete queue on the network interface device, the DMA controller pushing values onto the DMA command complete queue, the processor popping values off the DMA command complete queue.

3. The method of Claim 2, wherein the processor uses the DMA command complete queue to determine that the first and the second portions of data are both present in local memory on the network interface device.

4. The method of Claim 1, wherein the network interface device comprises queue manager hardware, the queue manager hardware maintaining the DMA

command queue in static random access memory (SRAM), wherein the processor, the queue manager hardware, the SRAM, and the DMA controller are all part of the same integrated circuit.

Claim 5 is allowed and is not involved in the appeal.

6. The method of Claim 1, wherein each of the values on the DMA command queue is a DMA command.

7. The method of Claim 1, wherein each of the values on the DMA command queue comprises a pointer to a DMA command.

8. The method of Claim 1, wherein each of the values on the DMA command queue is a number that identifies a location where a DMA command is stored.

9. The method of Claim 1, wherein the network interface device is an expansion card coupled to a host computer, the host storage being part of the host computer.

10. The method of Claim 1, wherein the network interface device is a part of a host computer.

11. A network interface device, comprising:

queue manager hardware that maintains a DMA command queue;

a processor coupled to the queue manager hardware, the processor causing values to be pushed onto the DMA command queue;

a DMA controller coupled to the queue manager hardware, the DMA controller executing DMA commands, the DMA commands executed being indicated by values popped off the DMA command queue;

local memory that temporarily stores a first portion of data transferred by execution of one or more DMA commands from data storage on a host coupled

to the network interface device into the local memory, the local memory also temporarily storing a second portion of data transferred by execution of one or more DMA commands from the data storage on the host to the local memory; and

physical layer interface and media access control circuitry, the physical layer interface and media access control circuitry outputting the first and second portions of data from the network interface device to a network, the first and second portions of data being output in the form of a data payload of a network communication.

12. The network interface device of Claim 11, wherein the local memory is dynamic random access memory (DRAM).

13. The network interface device of Claim 12, wherein the queue manager hardware stores at least part of the DMA command queue in static random access memory (SRAM).

14. A method, comprising:

(a) maintaining on a network interface device a DMA command queue, the DMA command queue being maintained by queue manager hardware on the network interface device;

(b) a processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command;

(c) popping a value off the DMA command queue, a DMA controller on the network interface device then executing a DMA command indicated by the popped value;

(d) repeating (b) and (c) such that a first portion of data is transferred from a first place on the network interface device to a second place on the network interface device, and such that a second portion of data is transferred from the

first place on the network interface device to the second place on the network interface device;

(e) after both the first portion and the second portion have been transferred to the second place in (d), the processor taking a software branch; and

(f) after taking the software branch, the processor outputting the first and second portions of data from the network interface device.

15. The method of Claim 14, wherein the processor outputs the first and second portions of data in (f) to a network.

16. The method of Claim 14, wherein the processor outputs the first and second portions of data in (f) to a host computer.

17. The method of Claim 14, wherein the first place is a dynamic random access memory (DRAM) and wherein the second place is a bus interface.

18. The method of Claim 14, wherein the first place is a bus interface and wherein the second place is a dynamic random access memory (DRAM).

19. The method of Claim 14, wherein the first place is a bus interface and wherein the second place is a static random access memory (SRAM).

20. The method of Claim 14, wherein the first place is a static random access memory (SRAM) and wherein the second place is a bus interface.

21. A method, comprising:

(a) using a DMA command queue to ensure that a plurality of DMA moves are completed in a particular sequence, each of the DMA moves being a move of information from one location on a network interface device to another location on the network interface device, the DMA command queue being maintained by

queue manager hardware on the network interface device, the DMA moves being carried out by a DMA controller, the DMA controller being a part of the network interface device; and

(b) outputting at least part of the information from the network interface device.

22. The method of Claim 21, wherein the information output in (b) is output from the network interface device to a host computer, the host computer being coupled to the network interace device.

23. The method of Claim 21, wherein the information output in (b) is output from the network interface device to a network.

24. The method of Claim 21, wherein a first of the plurality of DMA moves is a move of at least a part of a frame of a session layer message, and wherein a second of the plurality of DMA moves is a move of at least a part of a subsequent frame of the session layer message.

Claims 25-27 are allowed and are not involved in the appeal.

28. An apparatus, comprising:

means for maintaining a DMA command queue on a network interface device and for causing values to be pushed onto the DMA command queue, each of the values pushed onto the DMA command queue being indicative of a DMA command; and

a DMA controller that executes the plurality of DMA commands such that the DMA commands are completed in a particular order.

29. The apparatus of Claim 28, wherein the means comprises a processor.

30. The apparatus of Claim 29, wherein the means further comprises a hardware queue manager.

31. A method, comprising:

(a) pushing values onto a DMA command queue in an order, each of the values being indicative of a different one of a plurality of DMA commands, wherein the DMA command queue is maintained on a network interface device (NID), and wherein the NID performs fast-path transport and network layer protocol processing; and

(b) popping the DMA command queue such that a DMA controller on the NID executes the plurality of DMA commands in the order in which the associated values were pushed onto the DMA command queue, the DMA controller being a part of the NID.

32. The method of Claim 31, wherein the NID includes a first memory and a second memory, the method further comprising:

(c) receiving multiple frames of a session layer network message onto the NID and storing the frames in the first memory, wherein execution of one of the plurality of DMA commands in (b) results in a move of at least a part of one of the frames from the first memory to the second memory, and wherein execution of another of the plurality of DMA commands in (b) results in a move of at least a part of another of the frames from the first memory to the second memory.

33. The method of Claim 31, wherein the NID is integrated into an integrated circuit taken from the group consisting of: a memory controller integrated circuit, a graphics controller integrated circuit, an input/output integrated circuit, and a bridge integrated circuit, and wherein the integrated circuit of which the NID is a part is realized on a motherboard of a host computer.

34. The method of Claim 31, wherein the NID is a means for performing fast-path transport and network layer protocol processing.

X. EVIDENCE APPENDIX

There is no evidence to submit in this evidence appendix. No evidence has been submitted pursuant to 37 C.F.R. §§ 1.130, 1.131 or 1.132.

XI. RELATED PROCEEDINGS APPENDIX

No decision has yet been rendered by a court or the Board in this or any related proceeding.